

# 10. Двоично представяне на числата

Проф. д-р Емил Хаджиколев

1. Бройни системи
2. Двоична бройна система
3. Операции върху двоични числа
4. Побитови оператори
5. Преобразуване на числа от една бройна система в друга

# Бройни системи

- Бройните системи са **метод за представяне на числа, включващ графични знаци и правила за записване на числата.**
- Една част от графичните знаци служат за означаване на цифри, а други (като десетична запетая) са спомагателни.

# Видове бройни системи

- Видове бройни системи (БС) – **позиционни** и **непозиционни**.
- При позиционните, стойността на цифрата зависи от мястото ѝ в числото, за разлика от непозиционните.
- **Примери:**
  - Непозиционна – римската бройна система;
  - Позиционна – арабската десетична бройна система.
- **Стойността на една цифра в позиционна БС зависи от позицията ѝ в записа на числото.** Напр. в числото 123, първата цифра 1 се възприема като сто, 2 като двадесет, а 3 – като три.
- Множителят, с който се изменя стойността на една цифра, се нарича **основа на бройната система**. Основата съвпада с броя използваните цифри.
- При едновременна работа с няколко бройни системи, за да не се допускат недоразумения, след всяко число, чрез долен индекс, се отбелязва бройната система. Напр.,  $84_{(10)}$ ,  $1010_{(2)}$ .

# Десетична бройна система

- **Десетичната бройна система** е с основа 10 и използва цифрите 0, 1, 2, 3, 4, 5, 6, 7, 8 и 9.
- Цифрите в записа на едно число имат различна стойност, която е степен на основата 10.
- Напр. в записа на цяло число, най-дясната цифра представя единиците ( $10^0$ ), следващата е за десетиците ( $10^1$ ), после за стотиците ( $10^2$ ) и т.н.
- Всеки разряд е 10 пъти по-голям от следващия го (в дясно) и 10 пъти по-малък от предшестващия го (от ляво).
- **Пример:**

$$123 = 1 * 10^2 + 2 * 10^1 + 3 * 10^0$$

# Двоична бройна система (1)

- В електрониката е лесно и евтино да се реализират логически схеми с две устойчиви състояния.
- Поради тази причина, в съвременните компютри се използва основно двоичната БС.
- Тя се състои от две цифри – 0 и 1 (означават съответно „няма ток“, „има ток“), чрез които може да се представи всякаква информация.
- За извършване на различни действия се използва двоична аритметика. Числата се четат от ляво на дясно – от старшия към младшия разряд.

# Двоична бройна система (2)

- При двоичната бройна система важат същите правила, като за десетичната бройна система.
- Използват се само цифрите 0 и 1 и всяка съседна цифрова позиция от ляво надясно е с нарастваща степен на числото 2.
- Напр.:

$$\begin{aligned} 110101_{(2)} &= 1*2^5 + 1*2^4 + 0*2^3 + 1*2^2 + 0*2^1 + 1*2^0 \\ &= 1*32 + 1*16 + 0*8 + 1*4 + 0*2 + 1*1 \\ &= 32 + 16 + 4 + 1 \\ &= 53_{(10)} \end{aligned}$$

# Записване на числа със знак в двоична БС

- В двоичен код може да се записват не само цели числа, но и числа със знак и дробни числа.
- За записване на числа със знак се използва специален **бит за означаване на знака**. Ако битът за знак има стойност 0, числото е положително, ако стойността му е 1 – числото е отрицателно.
- Така, ако за записа на цяло число се отделя 1 байт, в първия бит се записва знака, а в останалите 7 бита – самото число.
- Обхватът на целите числа със знак, които могат да се запишат в един байт е от  $-2^7$  до  $2^7-1$ , т.е. от -128 до 127.
- **Пример** за положително число:  $01001101_{(2)} = 77_{(10)}$ ,

# Прав и обратен код

- Положителните числа се представят в **прав код**.
- **Обратен код** – всички битове се инвертират (заменя се 0 с 1 и 1 с 0).
- Отрицателните числа се представят в **допълнителен код**, при който числото се прави в обратен код и към него се добавя 1.
- При ползването на двете кодирания за числата (в прав и допълнителен код) **нулата има само едно представяне**.
- **Пример:** За да получим  $-77_{(10)}$ , инвертираме побитово  $77_{(10)}$  и добавяме 1:  
$$\sim 01001101_{(2)} + 00000001_{(2)} =$$
$$10110010_{(2)} + 00000001_{(2)} =$$
$$10110011_{(2)} = -77_{(10)}$$
- Със символът ‘~’ се означава оператора побитово отрицание.

# Записване на дробни числа в двоична БС

- За запис на дробни числа се използва позиционна точка, която съответства на десетичната точка в математиката.
- Цифрите вляво от позиционната точка означават цялата част на числото, а тези вдясно – дробната част.
- Цялата част на числото се изчислява по разгледания вече начин.
- Дробната част се изчислява, като цифрите се умножават с отрицателни степени на числото 2 т.е. първата цифра след позиционната точка се умножава с  $2^{-1}$ , втората с  $2^{-2}$ , третата с  $2^{-3}$  и т.н.
- **Например:**

$$\begin{aligned} 1010.0110_{(2)} &= 1*2^3 + 0*2^2 + 1*2^1 + 0*2^0 + 0*2^{-1} + 1*2^{-2} + 1*2^{-3} + 0*2^{-4} \\ &= 8 + 2 + 1/4 + 1/8 = 10 + 3/8 = 10 \frac{3}{8} \end{aligned}$$

# Операции върху двоични числа

- **Аритметичните операции** за събиране и изваждане на двоични числа се извършват по същия начин, както и при десетичните – поразредно.
- **Операцията между  $i$ -тите разряди на двата операнда формира  $i$ -тия разряд на резултата.**
- Понякога има **пренос на единица към по-старшия разряд** при събиране и заемане от по-старшия разряд при изваждане.
- **Примери:**  
$$1001_{(2)} + 1101_{(2)} = 01110_{(2)}$$
$$10100_{(2)} - 1011_{(2)} = 1001_{(2)}$$

# Побитови операции

- С двоичните числа може да се извършват и **побитови операции** - отрицание ( $\sim$ ), "И" ( $\&$ ), "ИЛИ" ( $|$ ) и изключващо "ИЛИ" ( $\wedge$ ).
- Правилата са дадени в следващата таблица.

<b>a</b>	<b>b</b>	<b>NOT a</b> ( $\sim a$ )	<b>a AND b</b> ( $a \& b$ )	<b>a OR b</b> ( $a   b$ )	<b>a XOR b</b> ( $a \wedge b$ )
<b>0</b>	<b>0</b>	1	0	0	0
<b>0</b>	<b>1</b>	1	0	1	1
<b>1</b>	<b>0</b>	0	0	1	1
<b>1</b>	<b>1</b>	0	1	1	0

# Побитово отрицание

- **Побитовото отрицание** (NOT) се прилага върху един операнд и променя стойността на всеки бит на операнда.
- Битовете, които са имали стойност 0, се променят на 1, а тези със стойност 1, стават 0.
- Така се образува допълнението на даденото двоично число.
- **Пример:**  
 $\text{NOT } 110100 = 001011$

# Побитово „И”

- **Побитово „И” (AND)** има стойност 1, само ако и двата операнда имат стойност 1, в противен случай стойността му е 0.
- Може да се използва **да се провери стойността на даден бит.**
- **Например**, ако искаме да проверим каква е стойността на третия бит в числото 100110, може да извършим следната операция:

$$100110 \text{ AND } 000100 = 000100$$

Тъй като резултатът не е 0, това ще означава, че третият бит в нашето число е вдигнат, т.е. има стойност 1. Това се нарича **битово маскиране**. Операторът може да се използва да се „свалят” определени битове. За целта се използва втори операнд, който има стойност 0 във всички битове, които трябва да се свалят, а в останалите битове имат стойност 1.

# Побитово „ИЛИ”

- **Побитово „ИЛИ”** (OR) връща 0, само ако и двата операнда едновременно имат стойност 0, в противен случай връща 1.
- Може да се използва за „вдигане” на определени битове.
- **Например**, ако искаме да вдигнем 2 и 5 бит на числото 1000100, може да извършим следната операция:

$$1000100 \text{ OR } 0010010 = 1010110$$

# Побитово „ИЗКЛЮЧАЩО ИЛИ”

- **Побитово „ИЗКЛЮЧАЩО ИЛИ” (XOR)** връща резултат 0, ако двата бита имат една и съща стойност, и 1 – ако са с различна стойност.
- Операторът може да се използва за **сравняване на битове**.
  - **Например:**  $0010 \text{ XOR } 1000 = 1010$
  - Това означава, че 2-ри и 4-ти бит на двете числа са еднакви.
- Този оператор също може да е използва за **обръщане на битове**. За целта трябва да се използва операнд, който има стойност 0 в тези позиции на битовете на целевото число, които трябва да се обърнат.

# Побитово отместване

- Специфични побитови оператори са **побитово отместване вляво** (<<) и **побитово отместване вдясно** (>>).
- Те извършват **отместване наляво или надясно с указан брой позиции на всички битове** на едно число.
- Тези оператори изискват **два операнда** – левият е числото, върху което ще се извърши операцията, а десния указва броя на отместване на битовете.
- **Например:**  $1101001 \ll 3$  означава отместване наляво с 3 бита, като резултата ще е 1101001000, а  $1101001 \gg 3$  изисква отместване надясно с 3 бита, тук в резултат ще се получи 0001101.
- **Логически побитовото отместване вляво съответства на операцията умножение, а отместването вдясно – на деление.** Това е поради факта, че в записа на едно число всеки ляв бит има стойност, 2 пъти по-висока от бита отдясно т.е. преместване на 1 бит наляво умножава числото по  $2^1$ , преместване на 2 бита извършва умножение по  $2^2$ , на 3 бита –  $2^3$  и т.н.



# Побитови оператори - пример

```
#include <iostream>
#include <bitset> // библиотека за работа с битове
using namespace std;

int main() {
    int i = 20;

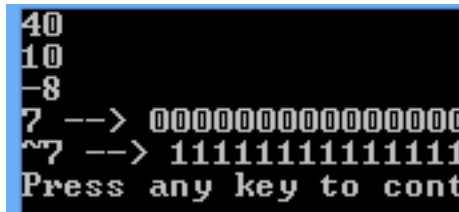
    i = i << 1; // изместване в ляво с един бит - това не е операторът за изход към конзолата
    cout << i << '\n';

    i = i >> 2; // изместване в дясно с два бита
    cout << i << '\n';

    cout << (~7) << '\n';

    cout << "7 --> " << bitset<32>(7) << '\n'; // представяме в 32 бита (int) числото 7
    cout << "~7 --> " << bitset<32>(~7) << '\n';

    return 0;
}
```



```
40
10
-8
7 --> 00000000000000000000
~7 --> 11111111111111111111
Press any key to continue
```

[illegible]

# Други бройни системи

- **Недостатък на двоичната бройна система** е дългият запис на цифрите.
- Често за представянето им, освен десетична се използват **осмична** и **шестнадесетична бройни системи**, поради факта, че 8 и 16 са степени на двойката (съответно  $2^3$  и  $2^4$ ).
- **Осмичната бройна система** използва цифрите от 0 до 7.
  - Напр.  $6021_{(8)} = 6 \cdot 8^3 + 0 \cdot 8^2 + 2 \cdot 8^1 + 1 \cdot 8^0 = 6 \cdot 512 + 0 + 16 + 1 = 3072 + 17 = 3089_{(10)}$
- **Шестнадесетичната бройна система** използва 16 символа – цифрите от 0 до 9, A, B, C, D, E и F. Първите 6 букви от английската азбука означават числата от 10 до 15, както следва: A = 10, B = 11, C = 12, D = 13, E = 14, F = 15.
  - Напр.  $6B2_{(16)} = 6 \cdot 16^2 + 11 \cdot 16^1 + 2 \cdot 16^0 = 6 \cdot 256 + 176 + 2 = 1714_{(10)}$

# Преобразуване на числа от една бройна система в друга

- Всяко число може да се **преобразува от една бройна система в друга.**
- В компютърните системи преобразуванията се налагат, за да може компютърните двоични числа да се представят в удобния за човека десетичен вид и обратно – въведените от хората десетични числа да се запишат в компютъра като двоични.

# Преобразуването на число от десетична в двоична бройна система

- Преобразуването на едно число от десетична в двоична бройна система, изисква деление на основата, в случая 2, като се записват последователно остатъците.
- Ако числото се дели на 2, се записва остатък 0, а ако не се дели – остатъкът е 1.
- След като деленето приключи, остатъците се записват в ред, обратен на реда, в който са получени и това е числото в двоична бройна система.

# Преобразуването на число от десетична в двоична бройна система - пример

- **Пример:** нека намерим как изглежда десетичното число 53 в двоична бройна система.

$$53 : 2 = 26, \text{ остатък } 1,$$

$$26 : 2 = 13, \text{ остатък } 0,$$

$$13 : 2 = 6, \text{ остатък } 1,$$

$$6 : 2 = 3, \text{ остатък } 0,$$

$$3 : 2 = 1, \text{ остатък } 1,$$

$$1 : 2 = 0, \text{ остатък } 1.$$

Така намираме, че  $53_{(10)} = 110101_{(2)}$ .

# Преобразуване на числа от една бройна система в друга

- Ако е необходимо да се извърши преобразуване на число от една бройна система в друга, при което **основата на едната бройна система е точна степен на другата**, може да се използва друг алгоритъм.
- **Пример:** преобразуване между двоична, осмична и шестнадесетична бройна система.

# Преобразуване от двоична в осмична БС - пример

Осмична	0	1	2	3	4	5	6	7
Двоична	000	001	010	011	100	101	110	111

*Таблица на съответствията между двете бройни системи*

- **Пример:** Да представим двоичното число 11011101 в осмична бройна система.
- Първо **разделяме двоичното число на тройки цифри**, като започнем отдясно на ляво.
- Ако най-лявата група се състои от по-малко от 3 цифри, **дописваме необходимия брой нули отляво**.
- За всяка тройка цифри търсим в таблицата съответната цифра в осмична бройна система:

$$11011101_{(2)} = 11|011|101 = 011|011|101 = 335_{(8)}$$

# Преобразуване между БС, чиито основи са точни степени на трето число

- Друг частен случай е, ако **основите на двете бройни системи са точни степени на трето число.**
- В този случай може да се използва **помощна бройна система** – с основа третото число.
- **Например**, осмичната и шестнайсетичната бройни системи са с основи, степени на числото 2 ( $8 = 2^3$ ,  $16 = 2^4$ ).
- Преобразуването на число от едната бройна система в другата, изисква с помощта на таблици за съответствия, **числото първо да се преобразува в помощната бройна система, и след това в целевата.**

# Таблицы за съответствията между цифрите от двете бройни системи и помощната БС

<b>Осмична</b>	0	1	2	3	4	5	6	7
<b>Двоична</b>	000	001	010	011	100	101	110	111

<b>Шестна- десетична</b>	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
<b>Двоична</b>	000	001	010	011	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111

# Пример за преобразуване на число от осмична в шестнадесетична БС

- **Пример:** Да се запише осмичното число 528 в шестнадесетична бройна система.

$$536_{(8)} = 101|011|110_{(2)} = 101011110_{(2)} = 1|0101|1110_{(2)} = \\ = 0001|0101|1110_{(2)} = 15E_{(16)}$$

Осмична	0	1	2	3	4	5	6	7
Двоична	000	001	010	011	100	101	110	111

Шестна- десетична	0	1	2	3	4	5	6	7	8	9	A	B	C	D	E	F
Двоична	000	001	010	011	100	101	110	111	1000	1001	1010	1011	1100	1101	1110	1111